基于WTL框架的Windows开发最佳实践

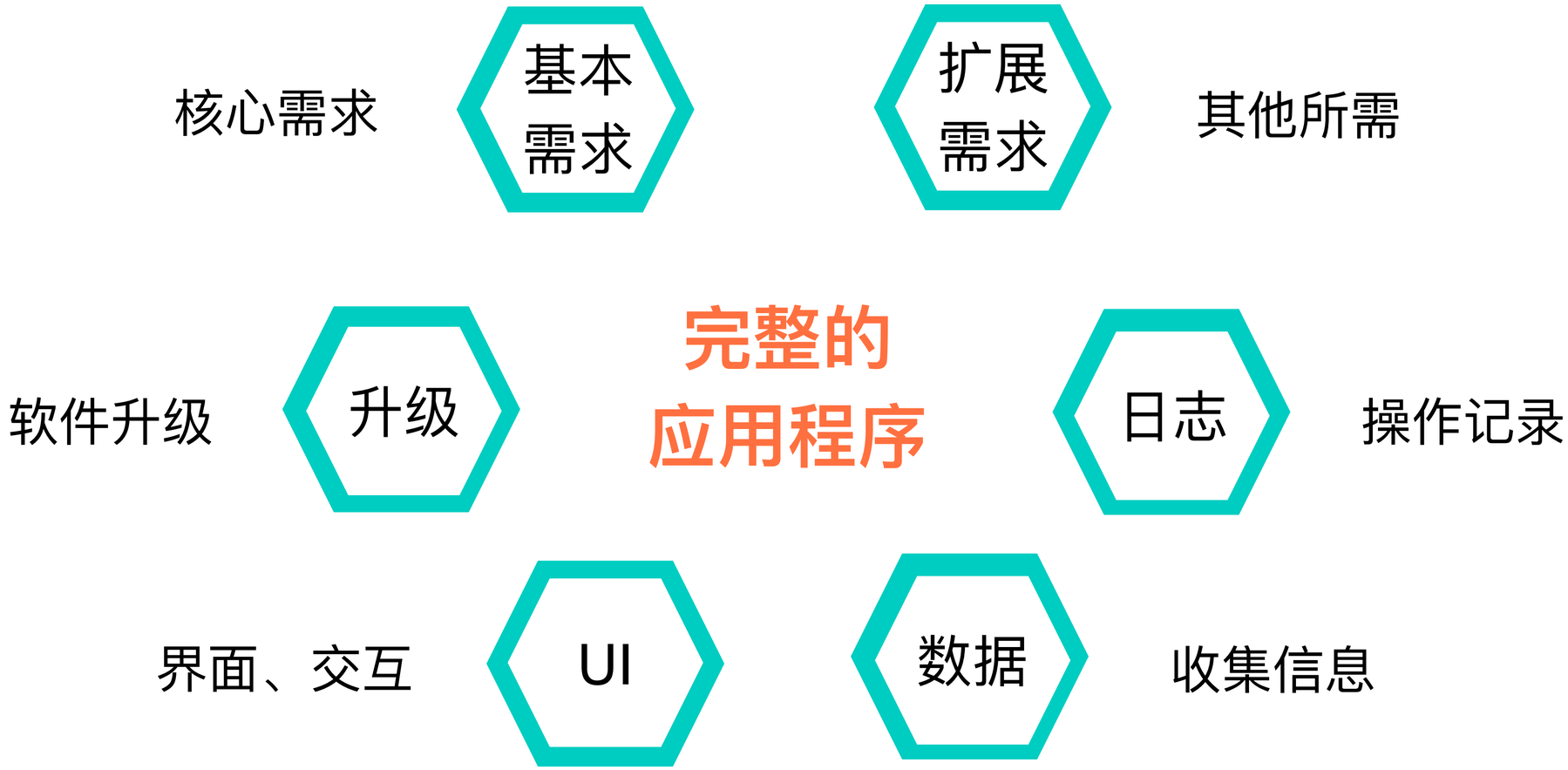
钱声鹏

美团点评

钱声鹏
美团点评高级工程师

擅长Windows客户端开发，目前参与多个客户端项目开发。

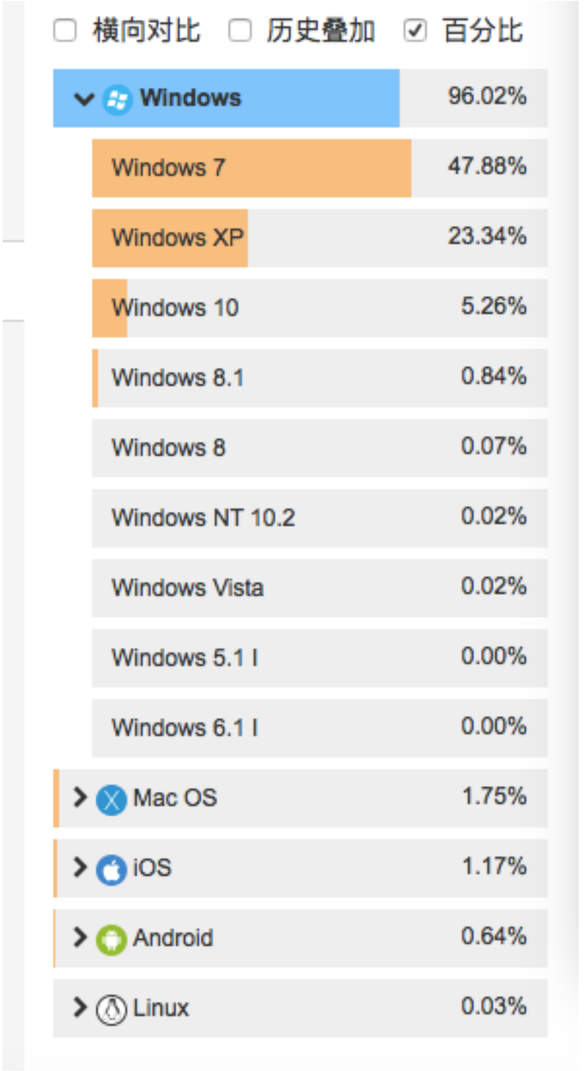
开发目标



用户诉求

环境现状

	商户PC特点
操作系统	Windows为主  
内存	不大，以2G和4G居多 
CPU	频率不高，以赛扬为主 



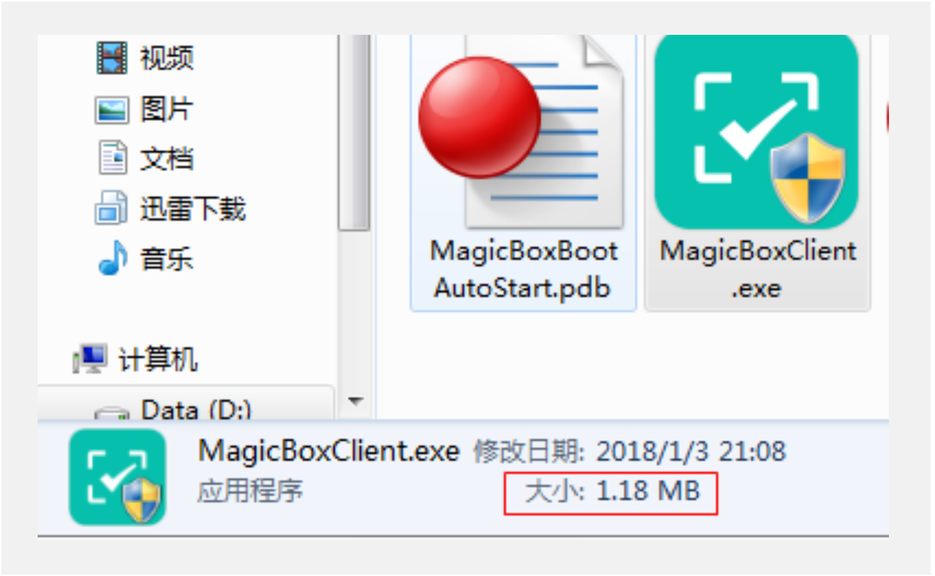
开发框架之争

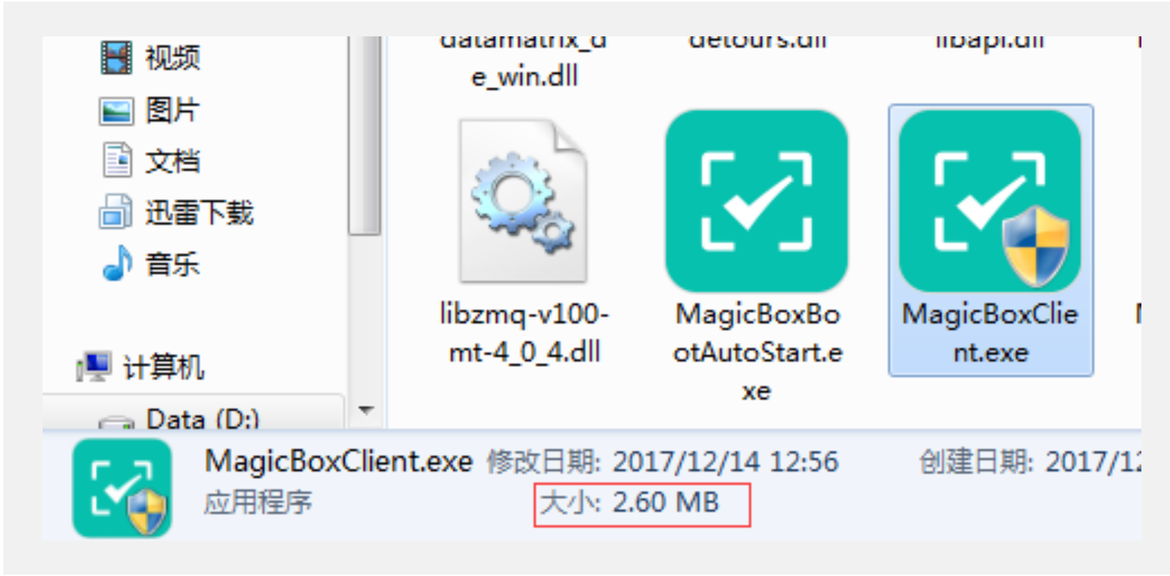
	WTL	MFC	WPF	QT
轻量	没有外部依赖 ★★★★★	依赖mfc.dll等库文件 ★★★★	需要安装.netframework ★	依赖全部的API和框架 ★★
开源	依赖Windows API ★★	依赖Windows API ★★	依赖.netframework ★	开源协议+商业协议 ★★★
控件	基本控件 ★	少量系统级高级控件 ★★	大量系统级高级控件 ★★★	大量高级控件 ★★★★★
性能	内存开销少、运行效率优 ★★★★★	内存开销较少 ★★★★	内存开销较多，运行效率一般 ★★★	内存开销较多，运行效率一般 ★★★
多平台	支持Windows各个平台 ★★★★	支持Windows各个平台 ★★★★	支持Windows各个平台 ★★★	支持Windows和Linux系统 ★★★★★
编码难度	难 ★	较难 ★★★	容易 ★★★★	容易 ★★★★

开发框架由需求决定

用户PC环境不佳、要求还高，所以**性能、轻量**就成为技术选型的重要指标；

小巧、轻量是WTL最大的特点。





性能对比

映像名称	CPU	工作设置(内存)
QQExternal.exe *32	00	54,820 K
igfxHK.exe	00	50,852 K
360DesktopLite64.exe	00	50,612 K
WeChatWeb.exe *32	00	50,456 K
大象.exe *32	00	48,928 K
jcef_helper.exe *32	00	46,916 K
360tray.exe *32	00	44,500 K
MagicBoxClient.exe *32	00	24,820 K
taskhost.exe	00	22,864 K
vpnui.exe *32	00	20,744 K
SogouCloud.exe *32	00	18,348 K
Paipaibox.exe *32	00	16,876 K
SoftMgrLite.exe *32	00	14,224 K
taskmgr.exe	00	11,868 K
RAVBg64.exe	00	11,844 K
RAVBg64.exe	00	11,784 K
RAVBg64.exe	00	11,764 K
RAVCpl64.exe	00	11,700 K

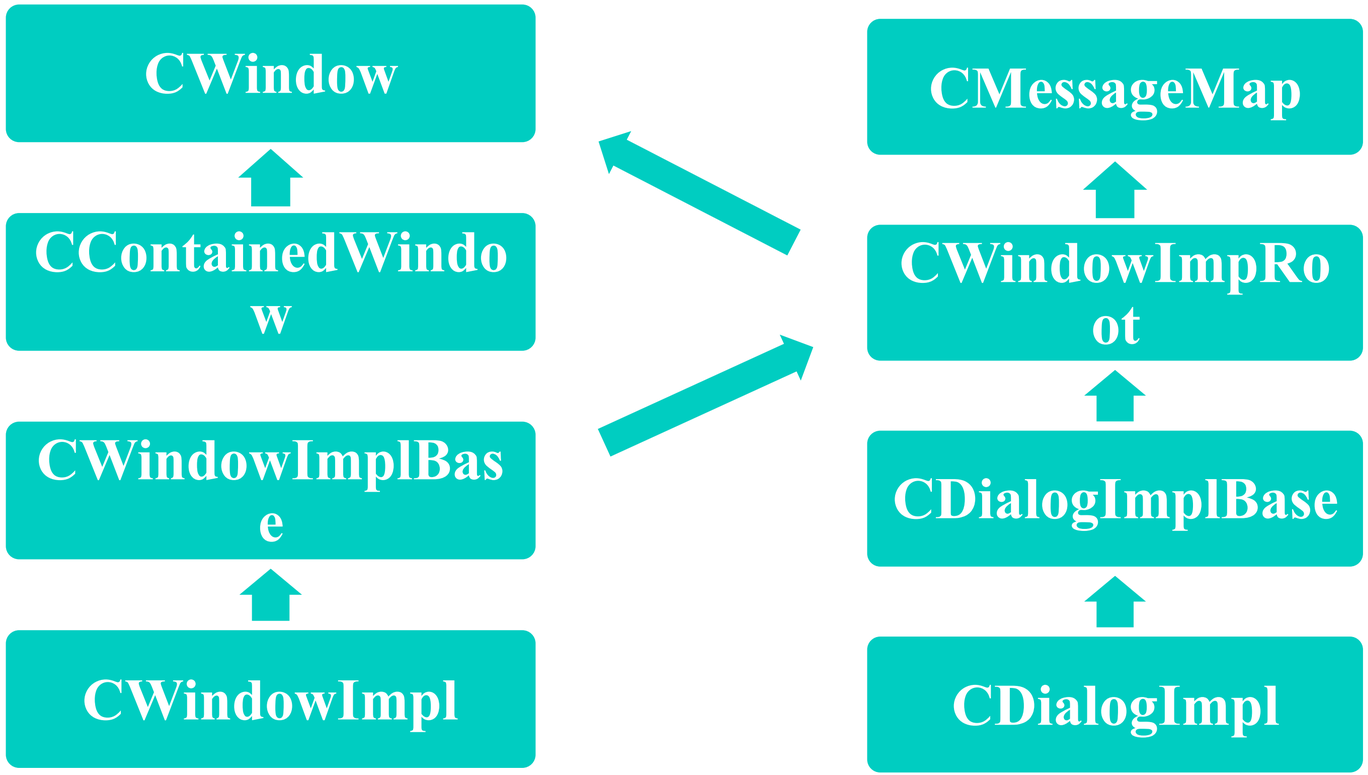
映像名称	CPU	工作设置(内存)
大象.exe *32	00	229,152 K
QQ.exe *32	00	227,636 K
WeChat.exe *32	00	182,052 K
大象.exe *32	00	128,972 K
云店助手.exe *32	00	96,220 K
explorer.exe	00	92,328 K
cloudmusic.exe *32	00	91,268 K
wpp.exe *32	00	86,116 K
csrss.exe	00	77,960 K
cloudmusic.exe *32	00	77,880 K
cloudmusic.exe *32	00	68,168 K
dwm.exe	00	60,948 K
WeChatWeb.exe *32	00	60,132 K
QQExternal.exe *32	00	54,844 K
igfxHK.exe	00	50,852 K
360DesktopLite64.exe	00	50,644 K
WeChatWeb.exe *32	00	50,456 K
大象.exe *32	00	48,928 K
jcef_helper.exe *32	00	46,916 K
360tray.exe *32	00	45,288 K
MagicBoxClient.exe *32	00	23,876 K
taskhost.exe	00	22,868 K
vpnui.exe *32	00	20,744 K
SogouCloud.exe *32	00	18,428 K
Paipaibox.exe *32	00	16,912 K

B扫C项目客户端运行时占用内存

WTL是什么

➢WTL基于ATL而上，采用C++模板技术包装了大部分窗口控制；

➢WTL对所有的Win32 通用对话框进行了封装：最基本的两个类——**CWindow、CMessageMap**



ATL是什么

➢ ATL是ActiveX Template Library （活动模板库）的缩写，它是一套C++模板库。

```
class BaseClass
{
      ......
};
class MyClass : public BaseClass
{
      ......
};
```

```
class  CMyWnd : public CWindowImpl<CMyWnd>
{
      ......
};
```

ATL与C++

```
class B1
{
public:
    void SayHi()      {  PrintClassName();  }
private:
    virtual void PrintClassName() { cout << "This is B1"; }
};

class D1 : public B1
{


};

main()
{
    D1 d1;    D2 d2;

    d1.SayHi();    // prints "This is B1"
    d2.SayHi();    // prints "This is D2"
}
```

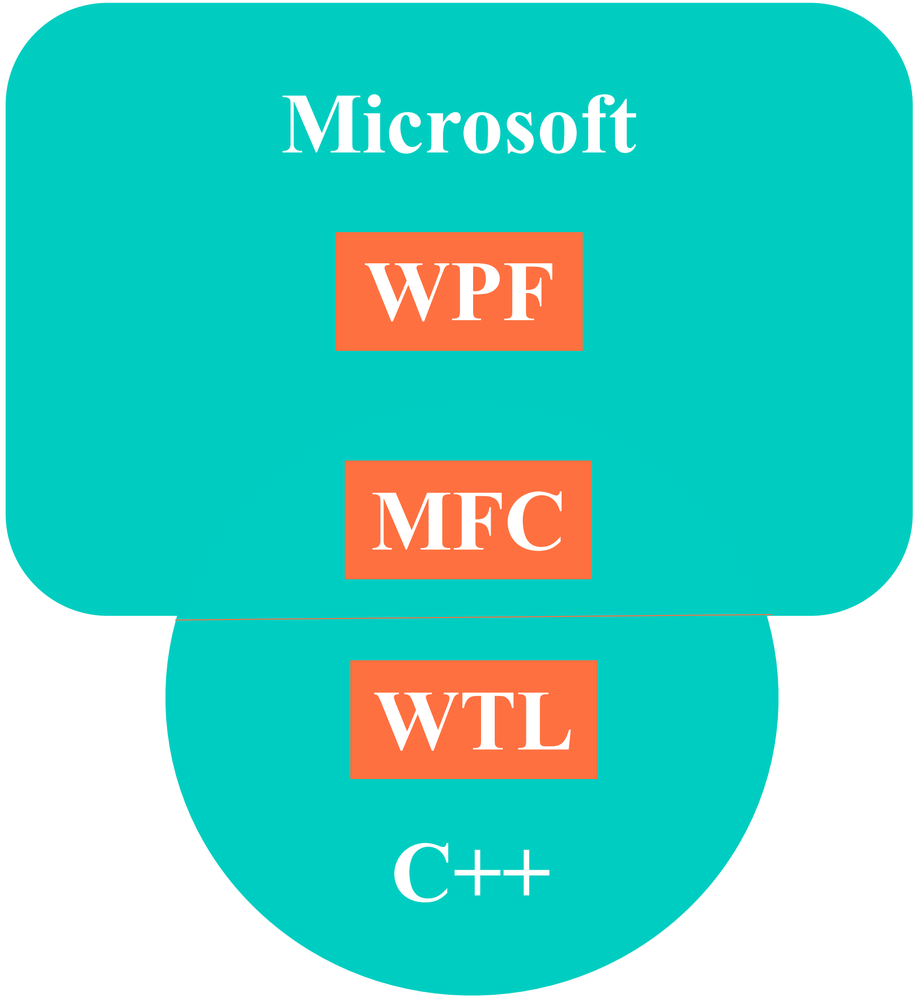
```
    class D2 : public B1
{
protected:
        void PrintClassName() { cout << "This is D2"; }
};
```

```
template <class T>
class B1
{
public:
    void SayHi()
    {
        T* pT = static_cast<T*>(this);
        pT->PrintClassName();
    }
protected:
    void PrintClassName() { cout << "This is B1"; }
};

class D1 : public B1<D1>
{


};

main()
{
    D1 d1;    D2 d2;

    d1.SayHi();    // prints "This is B1"
    d2.SayHi();    // prints "This is D2"
}
```

- **static_cast<T*>(this)** 就是窍门所在。它根据函数调用时的特殊处理将指向B1类型的指针this指派为D1或D2类型的指针；
- **节省内存，** 因为不需要虚函数表；

```
class D2 : public B1<D2>
{
protected:
    void PrintClassName() { cout << "This is D2"; }
};
```

WTL与MFC



MFC是微软提供的Windows下应用程序的编程语言接口，是一种软件编程的规范；

MFC是Win API与C++的结合，是对Win API的封装。

WTL是微软ATL开发组成员开发，内部使用；主要基于ATL对Win32 API的封装；

WTL与MFC

A

MFC和WTL对同一个事物定义不一样

MFC封装为类

WTL定义为结构体或者句柄

例如DC：MFC封装了CDC类，而WTL定义了HDC的句柄变量。

B

MFC和WTL对同一个过程定义不一样。

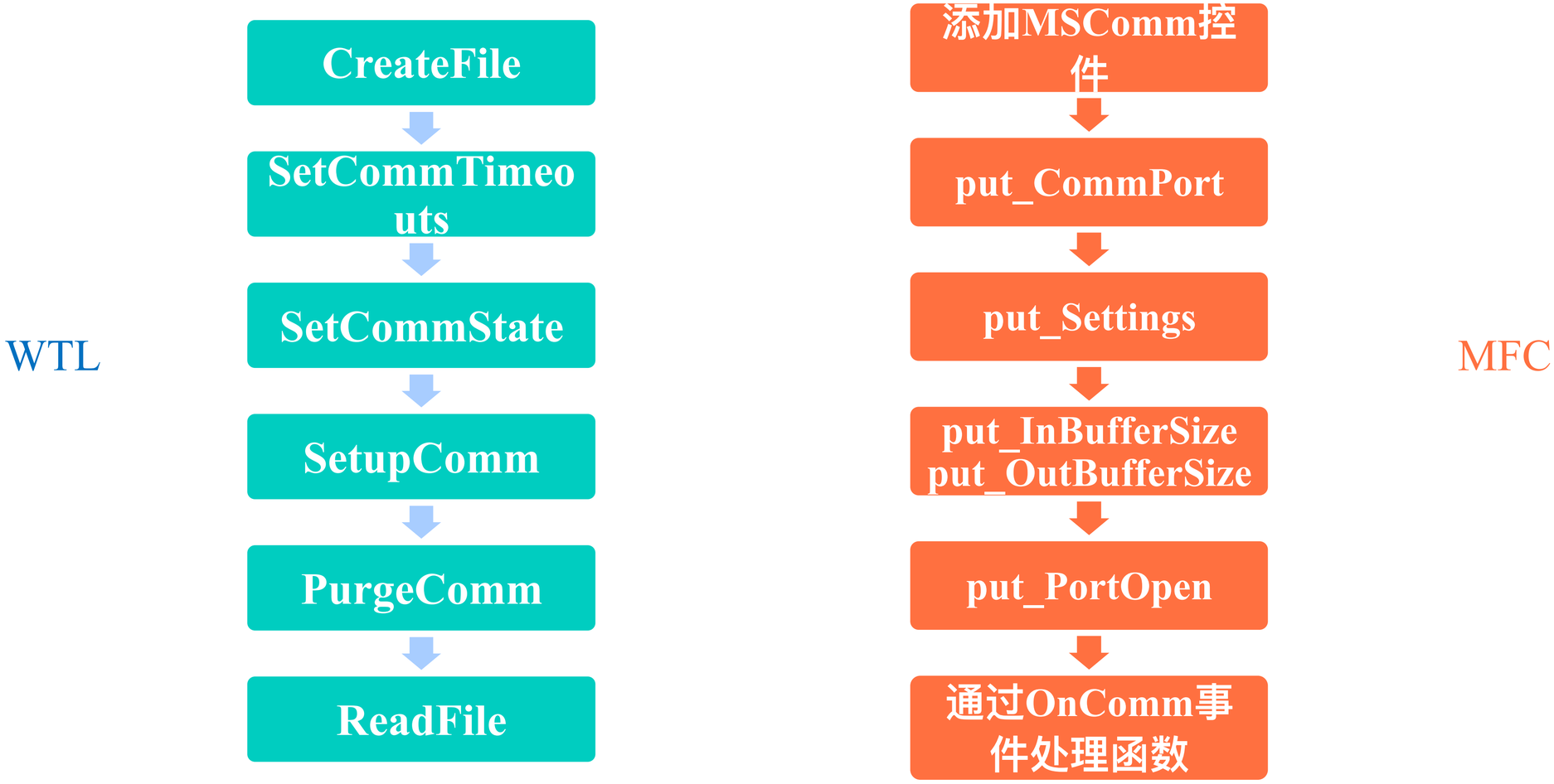
MFC封装成控件

WTL主要调用Win API

例如串口通信：MFC定义了MSComm控件，而WTL使用Win API

WTL与MFC

以串口通信编程为例



WTL与MFC

同一类

CString类在MFC和WTL中同时存在、用法也几乎相同，那在两个框架下，这个类是否是**一样实现** 吗?

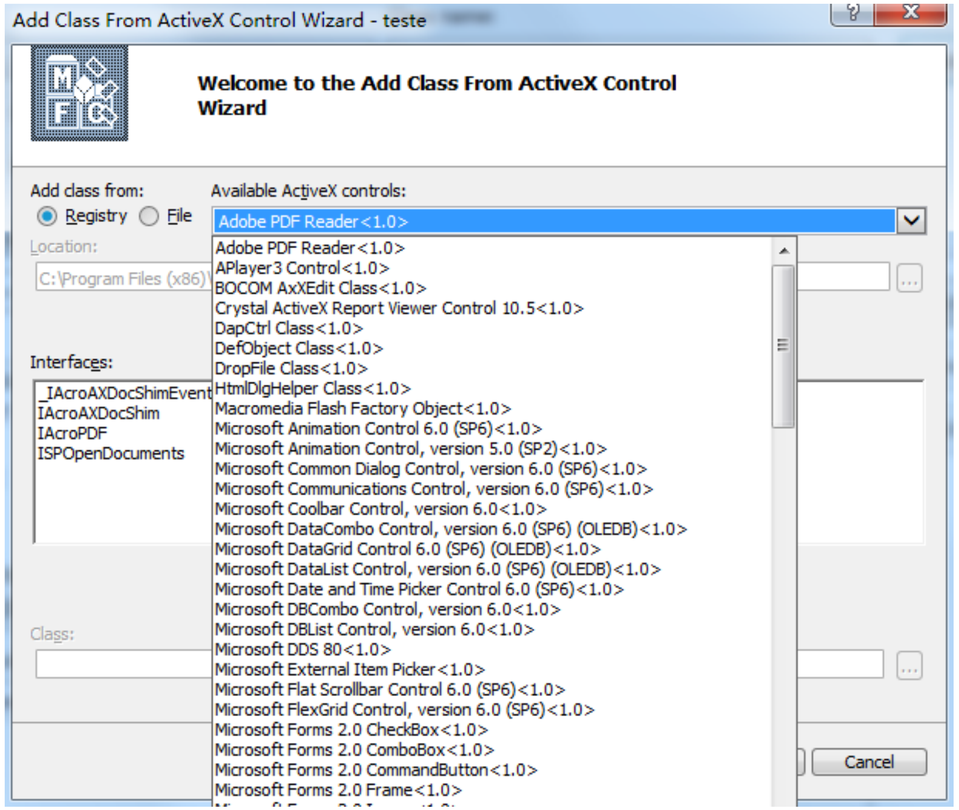
MFC

```
typedef ATL::CStringT< wchar_t, StrTraitMFC_DLL< wchar_t > > CStringW;
typedef ATL::CStringT< char, StrTraitMFC_DLL< char > > CStringA;
typedef ATL::CStringT< TCHAR, StrTraitMFC_DLL< TCHAR > > CString;
#else

typedef ATL::CStringT< wchar_t, StrTraitMFC< wchar_t > > CStringW;
typedef ATL::CStringT< char, StrTraitMFC< char > > CStringA;
typedef ATL::CStringT< TCHAR, StrTraitMFC< TCHAR > > CString;

#endif  // !_WIN64 && _AFXDLL
```

WTL

```
#ifndef _ATL_CSTRING_NO_CRT
typedef CStringT< wchar_t, StrTraitATL< wchar_t, ChTraitsCRT< wchar_t > > > CAtlStringW;
typedef CStringT< char, StrTraitATL< char, ChTraitsCRT< char > > > CAtlStringA;
typedef CStringT< TCHAR, StrTraitATL< TCHAR, ChTraitsCRT< TCHAR > > > CAtlString;
#else  // _ATL_CSTRING_NO_CRT
typedef CStringT< wchar_t, StrTraitATL< wchar_t > > CAtlStringW;
typedef CStringT< char, StrTraitATL< char > > CAtlStringA;
typedef CStringT< TCHAR, StrTraitATL< TCHAR > > CAtlString;
#endif  // _ATL_CSTRING_NO_CRT
```

WTL的不足



- 缺乏官方维护
- 界面Windows原生
- 编程依靠Win API
- 没有向导支持

对话框

WTL编程之道

创建（静态）一个对话框需要做**四件事**

01 创建一个对话框资源

02 从CDialogImpl类派生一个新类

03 添加一个公有成员变量IDD
将它设置为对话框资源的ID

04 建立窗口的消息映射

```
#pragma once

class CSubPageActivateAccount : public CDialogImpl<CSubPageActivateAccount>
{
public:
    enum { IDD = IDD_SUBPAGE_ACTIVATEACCOUNT };

    BEGIN_MSG_MAP(CSubPageActivateAccount)
        MESSAGE_HANDLER(WM_INITDIALOG, OnInitDialog)
        MESSAGE_HANDLER(WM_DESTROY, OnDestroy)
        MESSAGE_HANDLER(WM_ERASEBKGND, OnEraseBkgnd)
```

还记得刚才ATL定义一个派生类的使用方法吗?

对话框

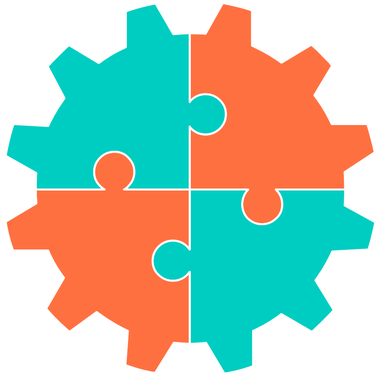
WTL编程之道

WM_INITDIALOG

对控件的初始化

WM_DESTROY

对控件的销毁



WM_ERASEBKGND

擦除背景

WM_PAINT

对控件的销毁

WM_SIZE

窗口大小发生改变，设置控件位置

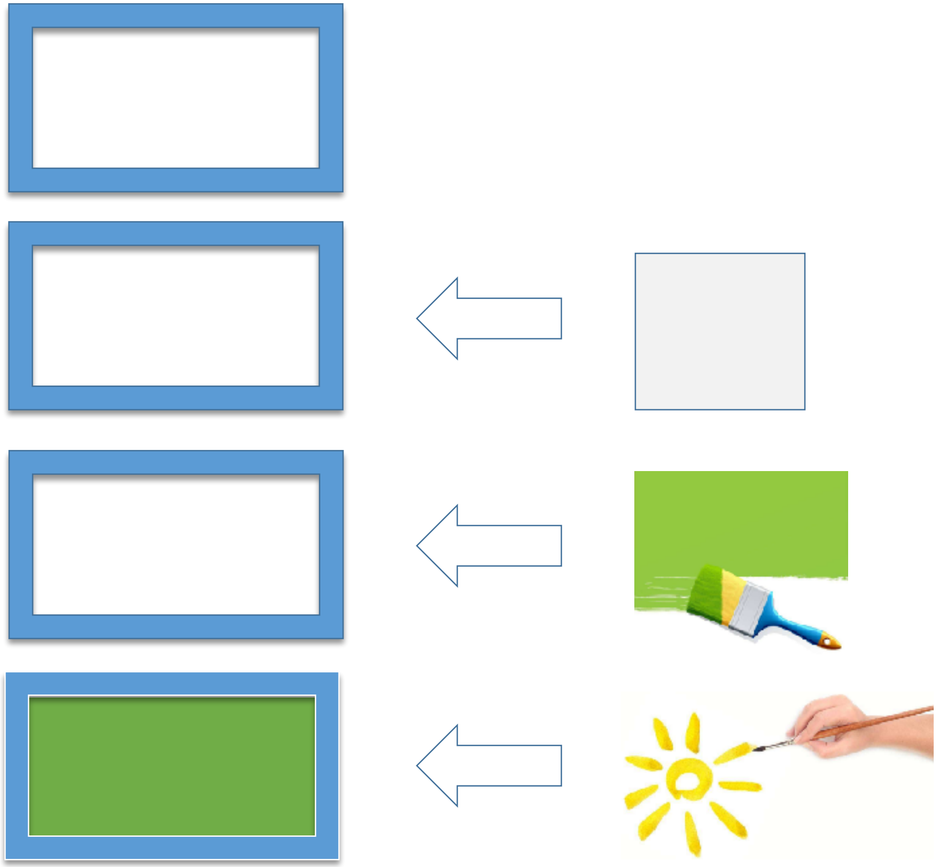
界面绘制

WTL编程之道

➢软件界面是人机交互的桥梁。用户体验的交互必需具有强烈的艺术效果，给用户留下深刻的第一印象，从而影响着客户的直接感观。

界面绘制

WTL编程之道



建立一个画板（HDC）

添加画纸（HBITMAP）

画底色（HBRUSH）

画内容

界面绘制

WTL编程之道

绘图流程

WM_PAINT（OnPaint）

……

绘制背景图（贴图）

设置透明

绘制文字

……

当绘制内容量很大的时候，或者界面频繁刷新的时候，还会出现闪烁现象。为了解决这个问题，采用**双缓冲技术**：

1.在内存中创建与画布一致的缓冲区

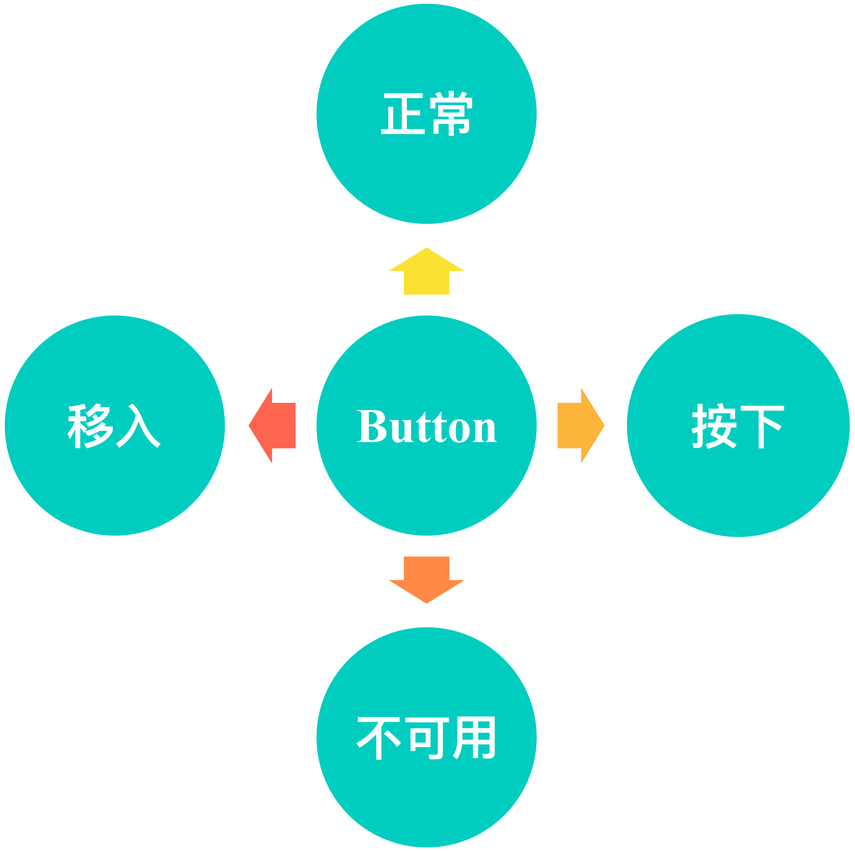
2.在缓冲区绘图

3.将缓冲区位图复制到当前画布上

4.释放内存缓冲区

界面绘制

WTL编程之道



➢ 解决移入状态变化：
WM_MOUSEMOVE（OnMouseMove）
WM_MOUSELEAVE（OnMouseLeave）

➢ 解决按下状态变化：
WM_LBUTTONDOWN（OnLButtonDown）
WM_LBUTTONUP（OnLButtonUp）

➢ 解决不可用变化：
SendMessage，将Disable状态通知控件

控件编写

WTL编程之道

需求

思路

对比

思路一

封装成List类和Item类，即每一行实为一个Item类，List类管理Item类的链表；

直接简单，但是开销大

思路二

封装成一个Window类，每一行实为一个结构体，该类管理结构体的链表；

实现难度稍大，但是开销少

控件编写

WTL编程之道

设计
内容结构体的公共变量

设计
公共方法

设计
按钮的效果、功能

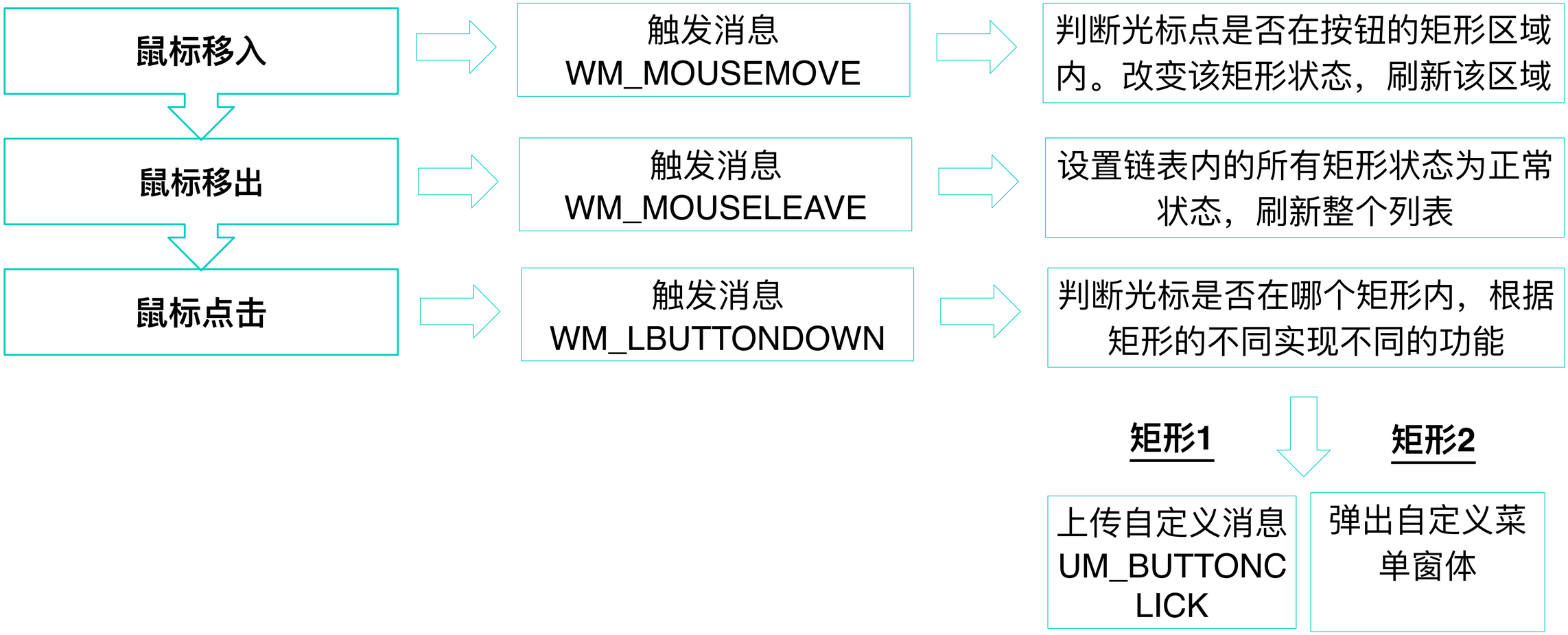
设计
滚动条

控件编写_公共方法

WTL编程之道

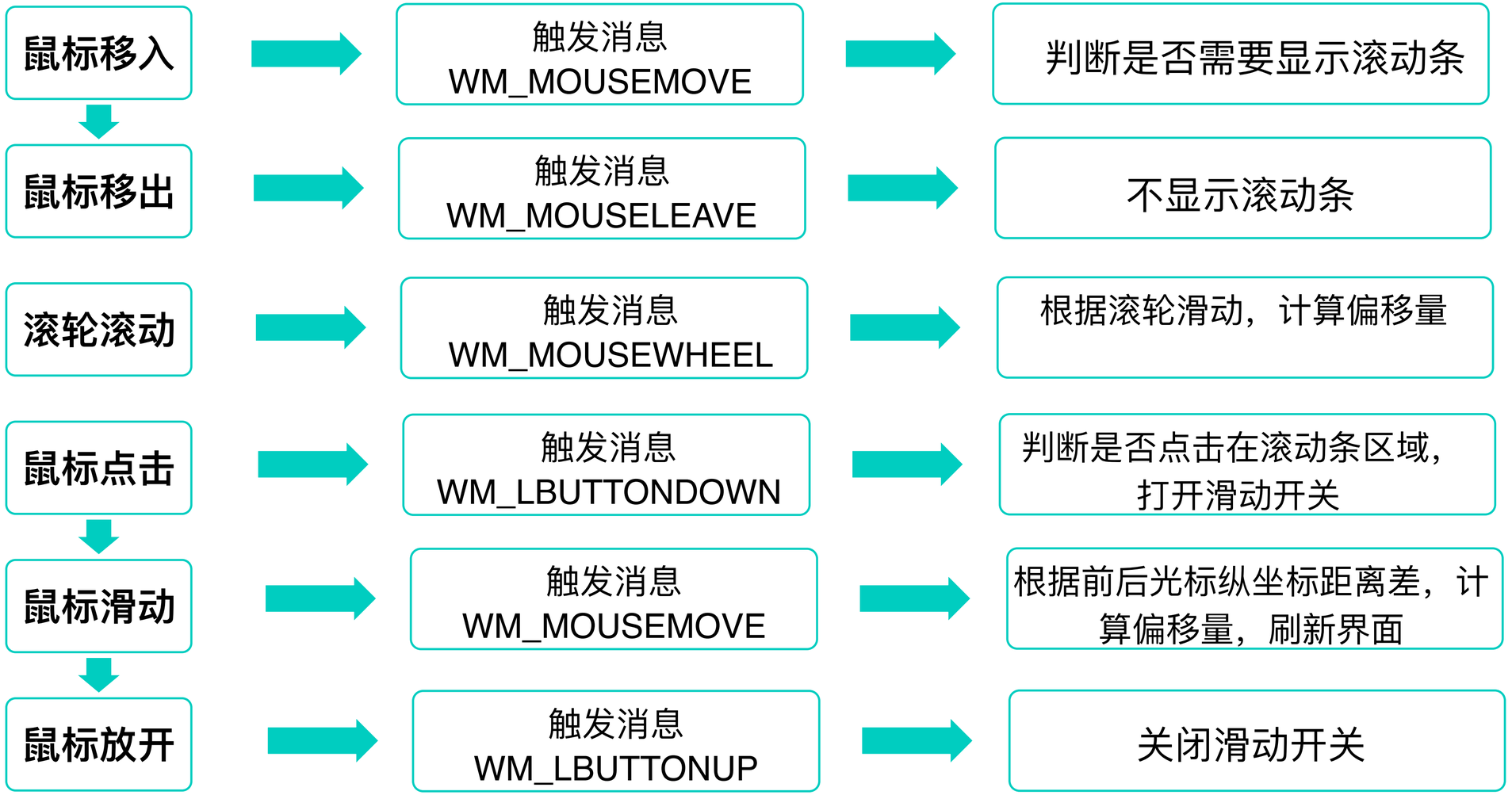
控件编写_按钮效果

WTL编程之道



控件编写_滚动条效果

WTL编程之道



总结

核心思想

不管是WTL、MFC还是QT，尽管各个框架的语法不同，但是核心都是C++，C++的核心思想不变的；

工作原理

不管使用什么语言，只要在系统的框架下运行，必须要对该系统的工作原理熟知；

解决问题

不管使用什么语言，目的都是为了高效解决问题，重点要是要掌握解决问题的思想；

更多技术资料
欢迎关注“美团点评技术团队”

招聘：Windows C++岗位
邮箱：qianshengpeng@meituan.com

美团点评